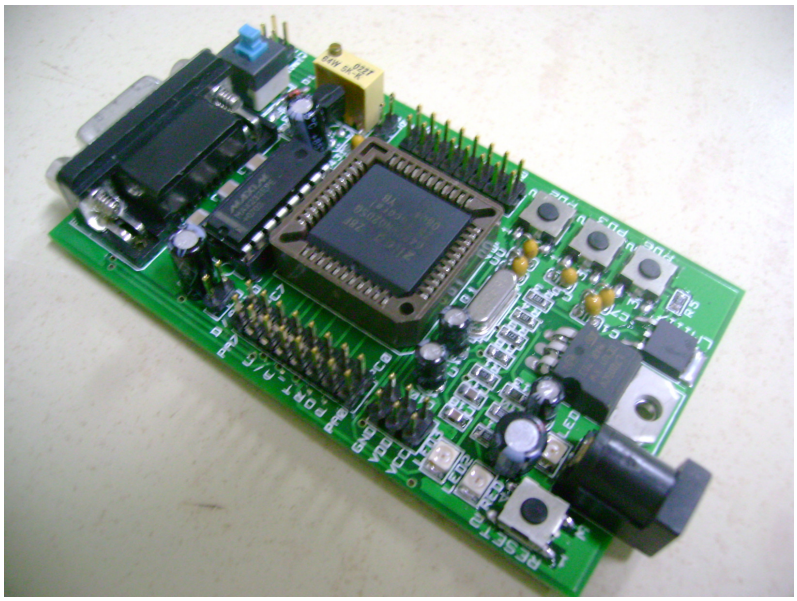


USER MANUAL

UP IRC SlimBoard set

INTRODUCTION

A microcontroller unit (MCU) or most commonly called as a microcontroller is a computer implemented on a single chip. Just like desktop and mainframe computers, the microcontroller also contains the basic components that these computers have such as the Central Processing Unit (CPU), Random Access Memory (RAM) and input/output devices (I/O) devices. However, microcontrollers are special purpose computers whose dedication is to perform one task at a time, unlike a desktop computer that can do several tasks simultaneously.

Microcontrollers are low-powered devices that are usually used by embedding them into other products or devices so that they can control the actions that these devices will do. It takes input from the device that it controls and also sends signals to it to manipulate its actions. A microcontroller needs to be programmed before it can be used to control devices.

Unlike a microprocessor that generally requires external components to implement a program, a microcontroller already incorporates all the resources needed internally in a single chip. Microcontrollers are usually designed to perform a small set of specific functions, whereas microprocessors tend to be designed to perform a wider set of general purpose functions that are performed in a personal computer (PC).

The UP IRC SlimBoard is a module that utilizes the on-chip peripherals of the Z8F6421 Flash Microcontroller, like the eight-channel Analog-to-Digital Converter (ADC) and the Universal Asynchronous Receiver/Transmitter (better known as UART). The board is powered using a 9VDC adapter and uses pin headers to access the input/output pins of Z8F6421 44 pin PLCC.

The Z8F6421 Flash Microcontroller is a Flash-capable microcontroller based on Zilog's advanced eZ8-8bit microprocessor core. The Flash in-circuit programming capability allows for faster development time and program changes in the field. Its rich peripheral set makes it suitable for a variety of applications including motor control, security systems, home appliances, personal electronic devices and sensors.

This User Manual contains guides in using the hardware and software materials needed in using the kit. It also includes instructions on how to install and use the ZDS II as a compiler and assembler plus examples and sample exercises to help the user understand the applications of the product.

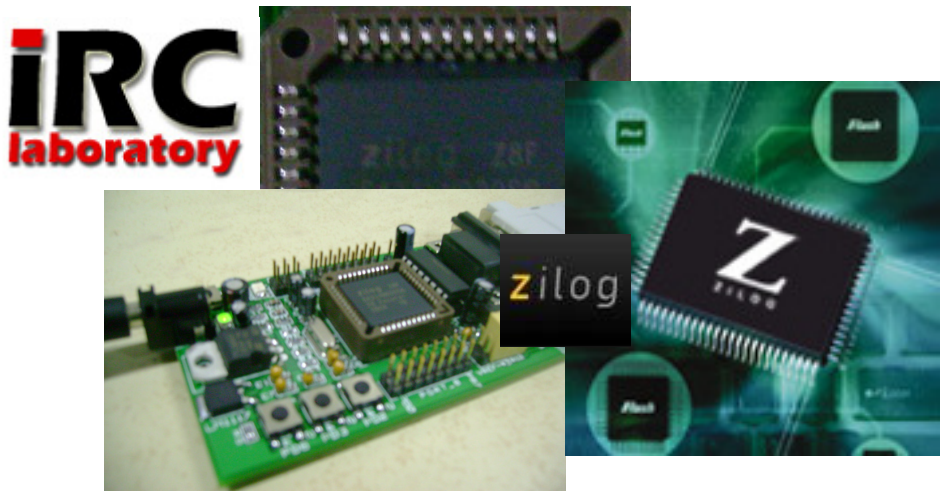
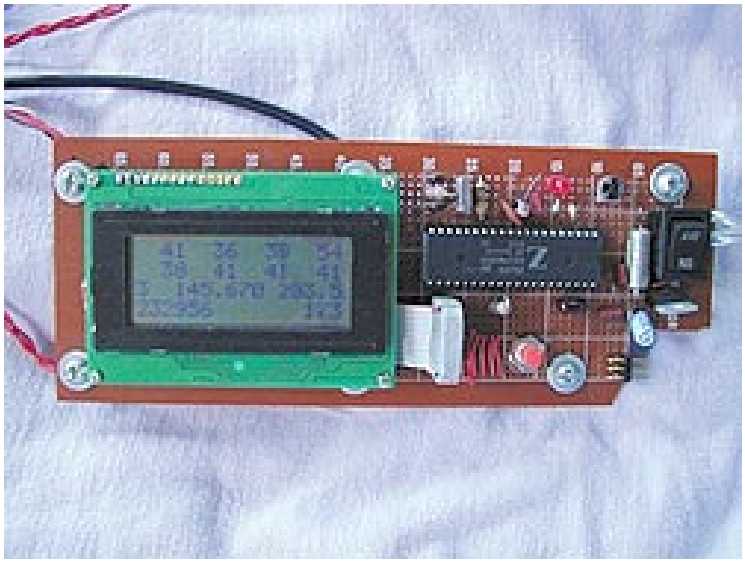


TABLE OF CONTENTS

Introduction 1
Table of Contents 2
Product Overview
- Product Features 3
- Product Additional Specifications 3
- UP IRC SlimBoard Lay-Out 4
- Package Contents 4

Before Getting Started
- Hardware Set-up 5
- ZiLOG ZDS II - Z8Encore! 4.10.1 and Hyperterminal 5

The C Language Structure 6
Getting Started
- Using the ZiLOG ZDS II - Z8Encore! 4.10.1 7
- Sample LED Blinker Code 18



An example of an application of Z8 Encore!
Ham Radio Repeater Locator

PRODUCT OVERVIEW

A. Product Features

The UP IRC SlimBoard set is designed to give electronic enthusiast an initial application material in their prototyping and laboratory needs. The board supports the Z8F6421 in the 44-pin package. These controllers have a program memory built on Flash technology which provides fast erasing and reprogramming, thus allowing a quick debugging or error checking. The UP IRC SlimBoard set has the following main features:

- eZ8 CPU 20MHz Operation
- 8-channel, 10-bit analog-to-digital converter
- 3-channel Direct Memory Access
- 64KB Flash memory with in-circuit programming capability
- 4KB register RAM
- Serial Communication protocols
- Two full-duplex 9-bit UARTs
- 24 interrupts with programmable priority
- Three 16-bit timers with capture, compare and PWM capability
- Single-pin On-Chip Debugger
- Two IrDA-compliant infrared encoder/decoders integrated
- Watch-Dog Timer with internal RC oscillator
- Up to 31 General Purpose Input/Output pins
- Voltage Brown-out Protection
- Power-On Reset
- 0 to +70 degree Celsius standard temperature operating ranges

B. Product Additional Specifications

The UP IRC SlimBoard set has the following specifications:

- 8 channel analog input (direct buffered) with GND header pair
- Adjustable external voltage reference
- PCB mounted DB9 RS-232 interface
- LED Power indicator
- +9V Unregulated Supply (from adaptor)
- +5V and +3.3V on-board regulation
- Remaining digital lines on headers

C. UP IRC SlimBoard Layout

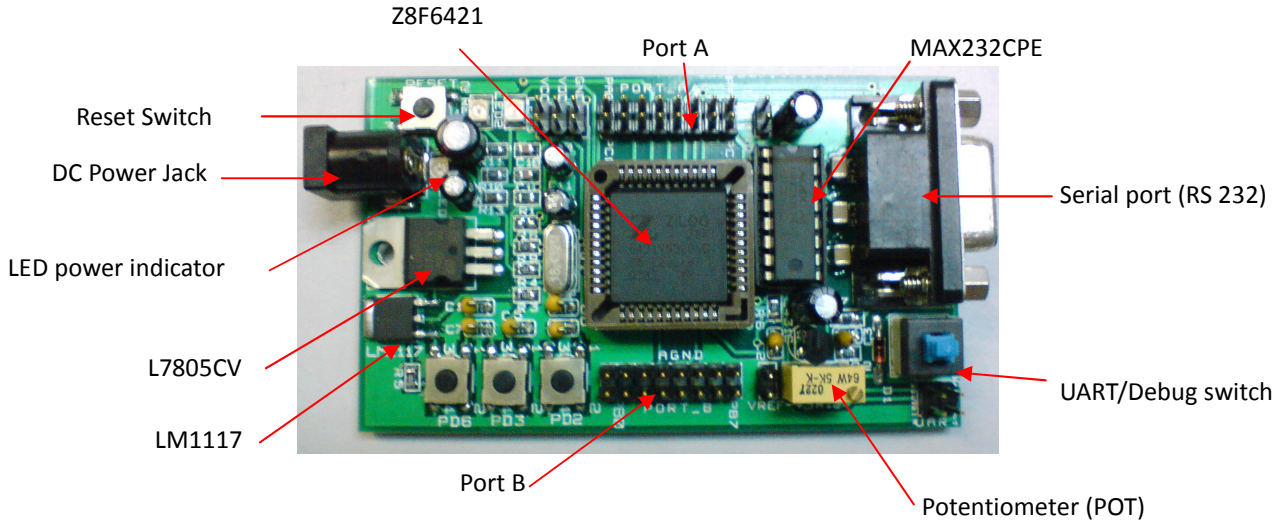


Figure 1. UP IRC SlimBoard Set

D. Package Contents

The UP IRC SlimBoard set should contain the following materials:

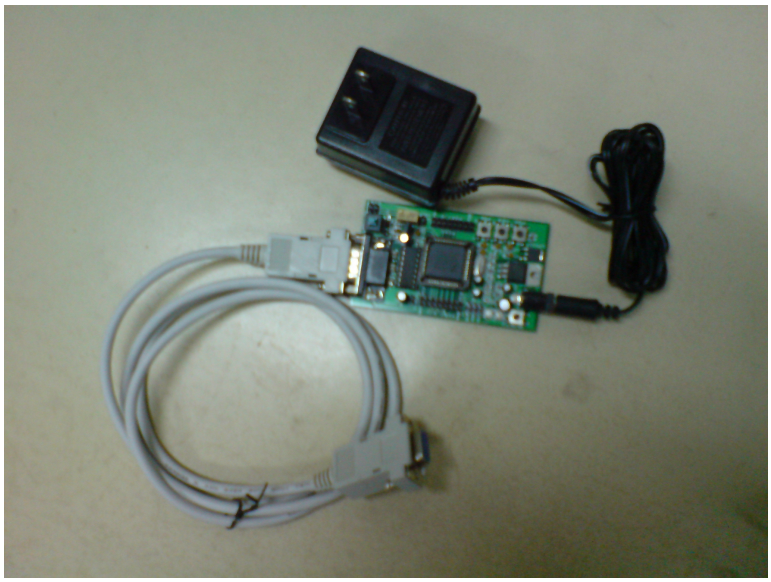
a. UP IRC SlimBoard with Z8F6421 Microcontroller

b. CD containing software tools, sample program codes and support documentations

c. Serial Cable (RS-232 data cable)

d. AC-DC power supply adapter

e. Portable case



BEFORE GETTING STARTED

A. Hardware Setup

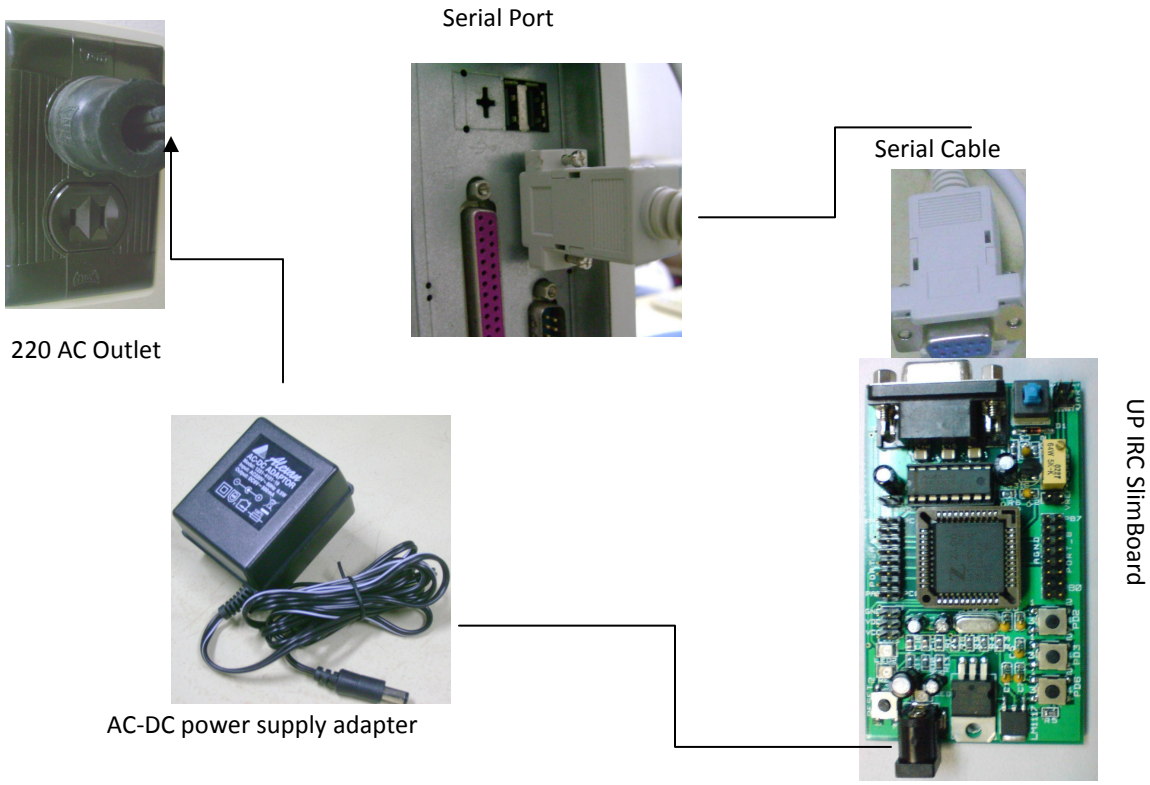


Figure3. UP IRC SlimBoard Set Hardware Setup

B. ZiLOG ZDS II - Z8Encore! 4.10.1 and Hyperterminal

A compiler is a software that is used to translate high-level languages (languages that are understood by humans) into machine-level languages. A program starts from a source code and is then translated by the compiler into a series of bits that tells the microcontroller what to do.

The **ZiLOG ZDS II – Z8Encore!** is a software program that can be used to easily compile and download the code into the SlimBoard. The HyperTerminal serves as the communication tool between the SlimBoard and PC.

THE C LANGUAGE STRUCTURE

C Language belongs to medium level of computer language which is the combination of high level language (human language) and low level language (machine language). In able to have a good program in C language, the algorithm or flow chart must establish first. This section features the structure of a C program.

Every program consists of one or more functions, one of which is called the *main*. This function is written as:

```
main() {
       expressions;
}
```

Additional function definition may precede or follow the main function. Each function must contain the following:

- A function heading, which must include the function name, followed by an optional list of arguments enclosed by parentheses.
- A list of argument declarations, if arguments are included in the heading.
- A compound statement, which comprises the remainder of the function.

Each compound statement is enclosed within a pair of braces, `{statements}`. The braces contain combinations of elementary or compound statements called *expressions*. Each expression must end with a semicolon `(;)`. *Comments* or remarks may appear anywhere in the program as long as they are placed within delimiters `(/*comments*/)` or double slash `(//)` in most microcontrollers. Comments play an important part in the documentation of the program.

Since all C programs must have their variables declared prior to using them, frequently used variables and references to predeclared C functions, may be used within a program can be contained within a separate *header file*. Header files can then declared prior to using them, frequently used variables and references to predeclared C functions, may be used within a program and be contained within a separate *header file*. Header files can then be combined into a program by calling them with an *#include* statement. All header files have the file extension of *.h*. The syntax of the header file is:

```
#include <header file> /*library file access*/
```

The program that you have written in C is called a *source code*. However, this cannot be executed on any physical hardware like the microcontroller in this form. It must be processed by the C Compiler and convert it into a HEX file. Shown below is the general structure of a C language.

```
#include <header file>     //declare header files to be used
#define statements         //declare the constants to be used

typedef and struct statements; //declare own type names and own
                                //instructions
global variables;                      //declare global variables
```

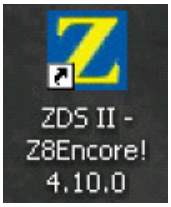
```
return_type user_function() {           //declare function name, return type
                                        //and parameters

local variables;
      statements;
return;                          //return value to the calling function
}

return_type main() {      //declare the return type and command line arguments
      local variables;
      code statements or expressions;
      }
```

GETTING STARTED

This sample program given below turns the LEDs on and off. This is why the program is called a “blinker”, which is one of the most common programs used to learn the basics of microcontroller programming. This program also prints “Hello World” in the HyperTerminal via the UART.

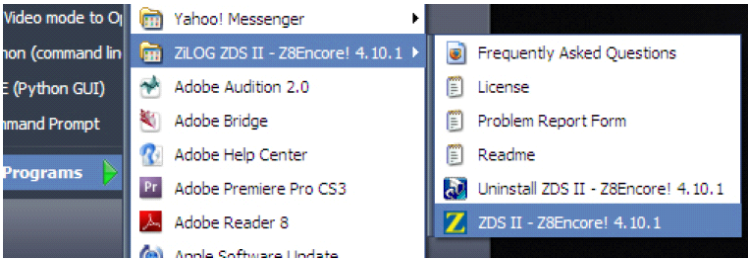
The implementation of this exercise is shown in the example below:

A. Using the ZiLOG ZDS II – Z8 Encore! 4.10.1

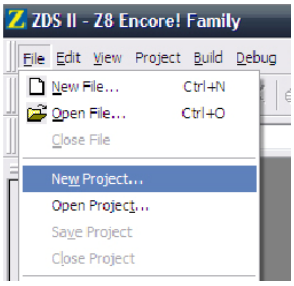


1. Creating a new project

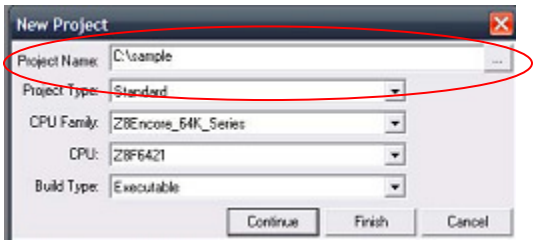
   a) Open **ZDS II.**



b) Click **File → New Project** from the main menu.



c) Enter **Project Name** and location.



d) Match this data:

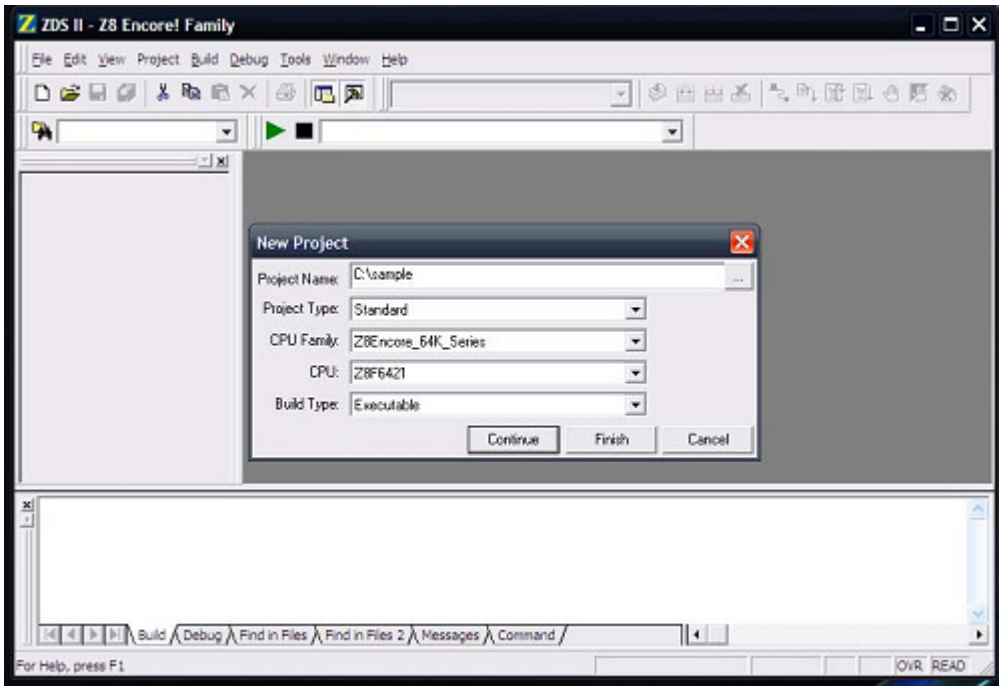
- *Since ZDS II can support many types of Z8 microcontrollers, we need to specify the specific kind of microcontroller.*

  i) Project type: **Standard**

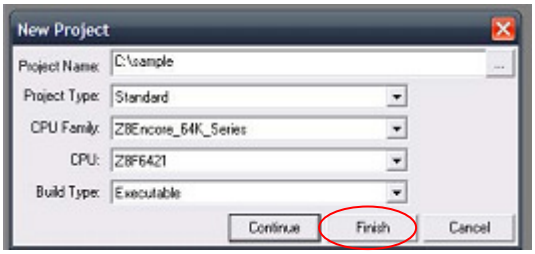
  ii) CPU Family: **Z8Encore_64K_series**

  iii) CPU: **Z8F6421**

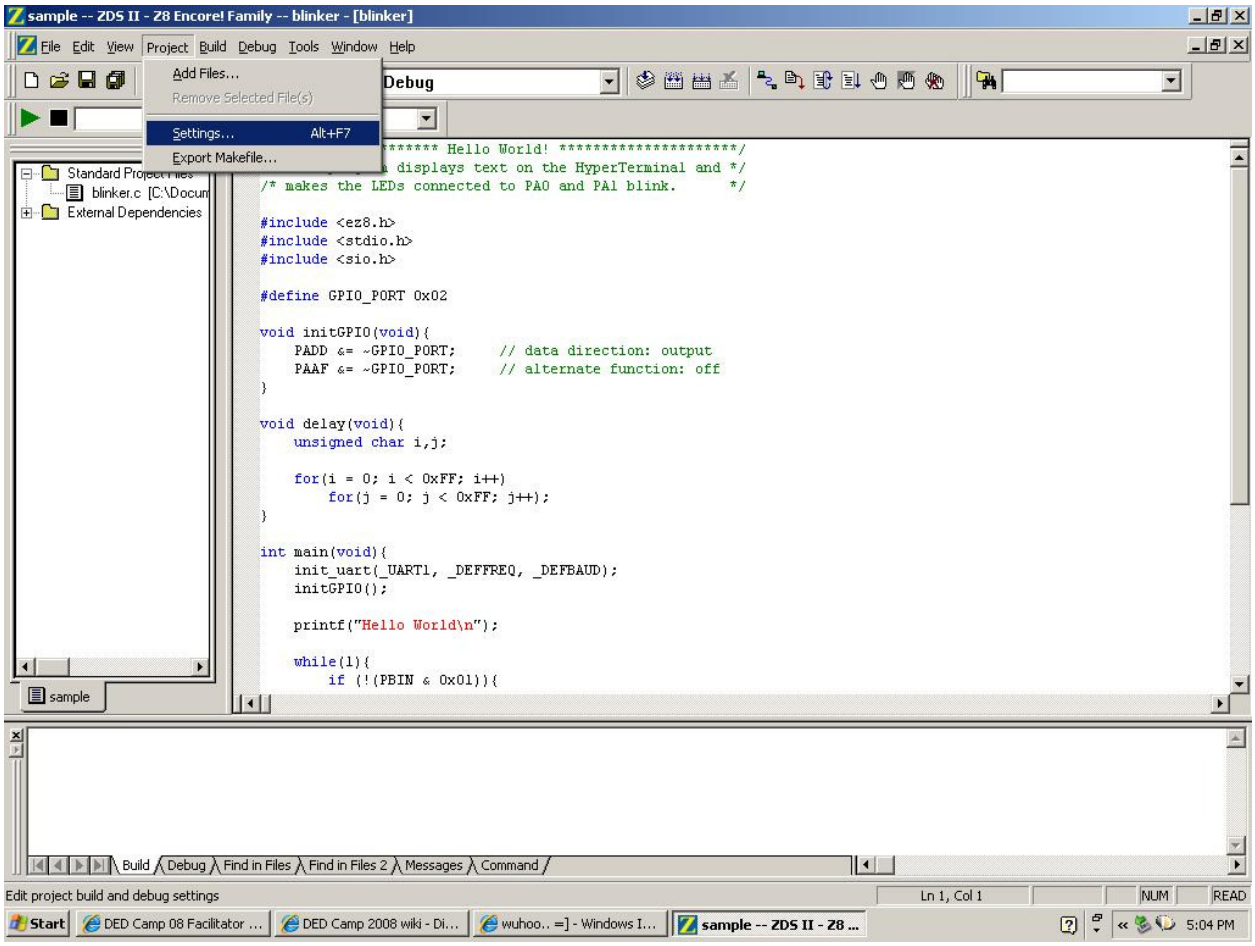
  iv) Build Type: **Executable**



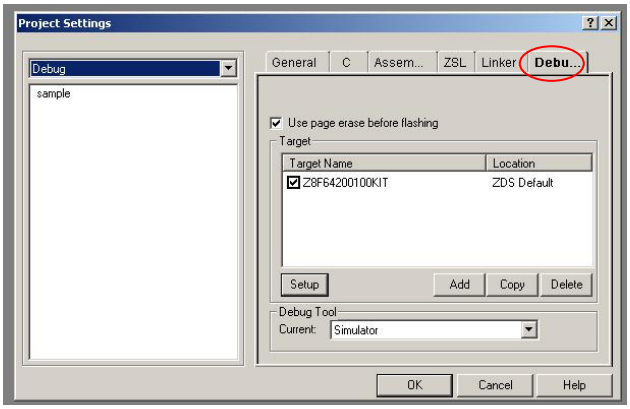
e) Click **Finish** button.



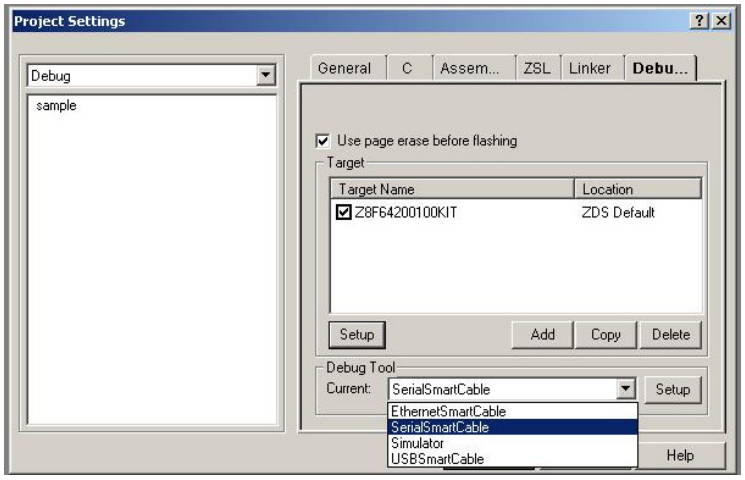
f) Click **Project → Settings** from the main menu.



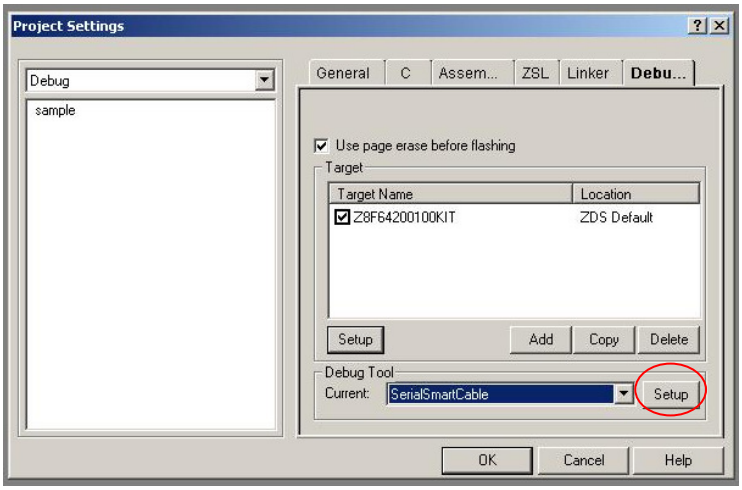
g) Click **Debugger** tab.



h) In debug tool, select **SerialSmartCable** from the dropdown menu.



i) Click **Setup** button.

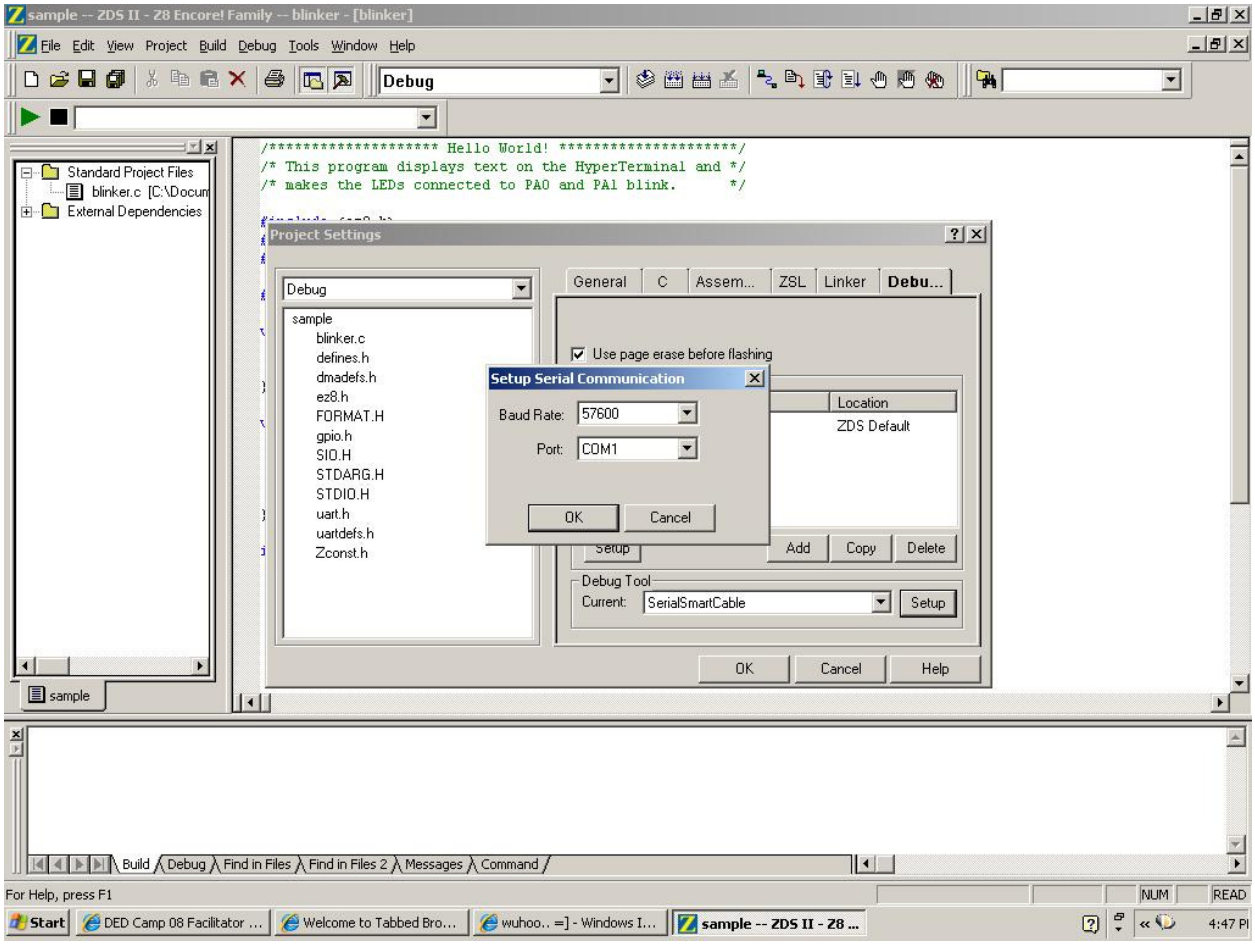


j) Match this data:

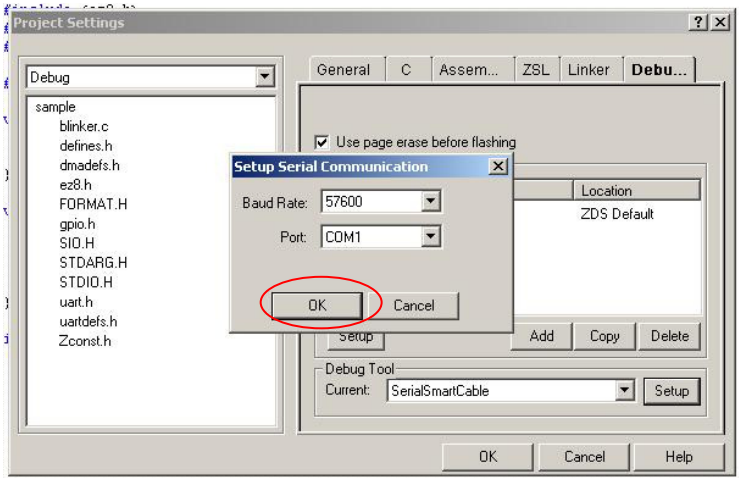
- *The setting for baudrate is the recommended setting for this particular microcontroller. In addition, there is a chance that there is a computer with more than one COM port. Therefore, we need to specify the correct COM port.*

  i) Baudrate: **57600**

  ii) COM port: select correct COM port



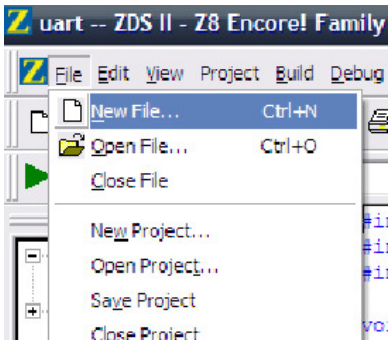
k) Click **OK** to return to default screen.



2. Adding a file to a project

● *Projects are different from files. Projects can have many files that contain the codes for the program. This can be header files or the main program, etc.*

a) When creating new file

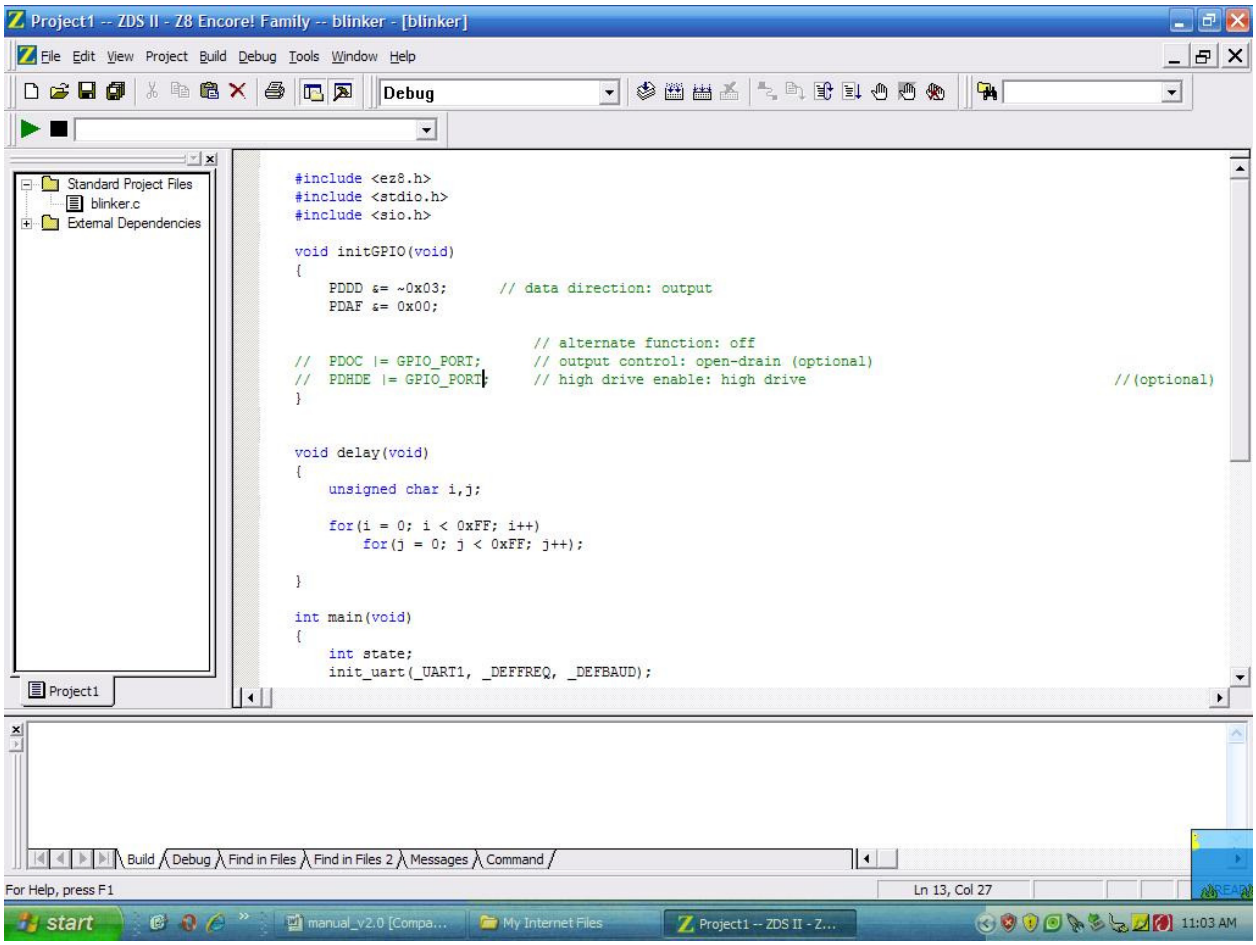


i) Click **File -> New File** from the main menu.

ii) Insert code

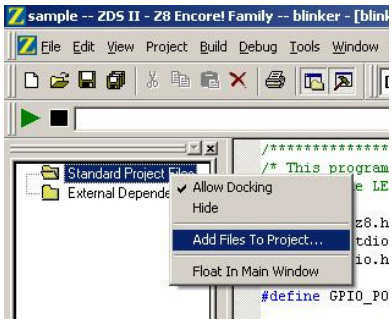
● *The code can be found at page 18. You can skip this step just to familiarize and test on how to proceed next.*

● Here is how it should look like:

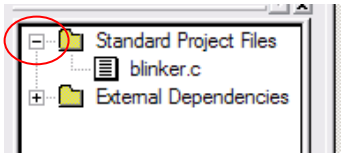


iii) Save file with ***.c** extension.

iv) Right click on the editor window, select **Add files to Project**.


sample -- ZDS II - Z8 Encore! Family -- blinker - [blink
File Edit View Project Build Debug Tools Window
Standard Project Files
External Depende
Allow Docking
Hide
Add Files To Project...
Float In Main Window
/* This program
#define GPIO_PO

v) To check, expand standard project files folder to see the file



b) When loading existing file to project

i) Right click standard project files folder , select **Add File** to project

ii) Select the file you want

iii) Double click file to show on the editor

3. Downloading code

a) Click **Rebuild All** on the toolbar

● *In this part, the program will compile your code. If there is any errors on your program, ZDS will not compile and displays the errors on the lower part of the window*

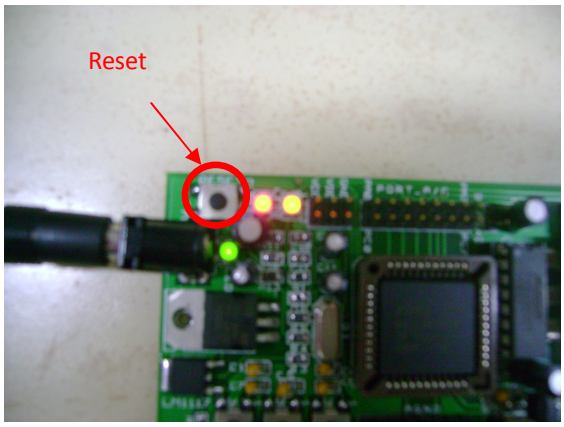
b) Click **Download Code** on the toolbar

● *You will know that the download is successful because it will displayed on the debug tab, seen on the lower part of the window*

c) When finished downloading, click stop debugging (shift+f5)

d) Push the **Reset** switch on the SlimBoard

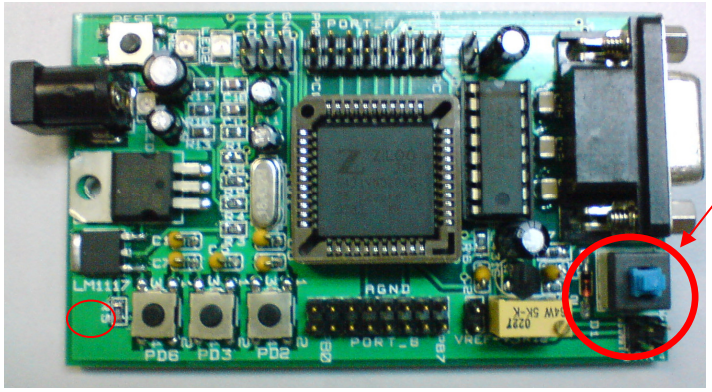
● *Look for the Reset button near the DC power jack. The display LEDs should "blink".*



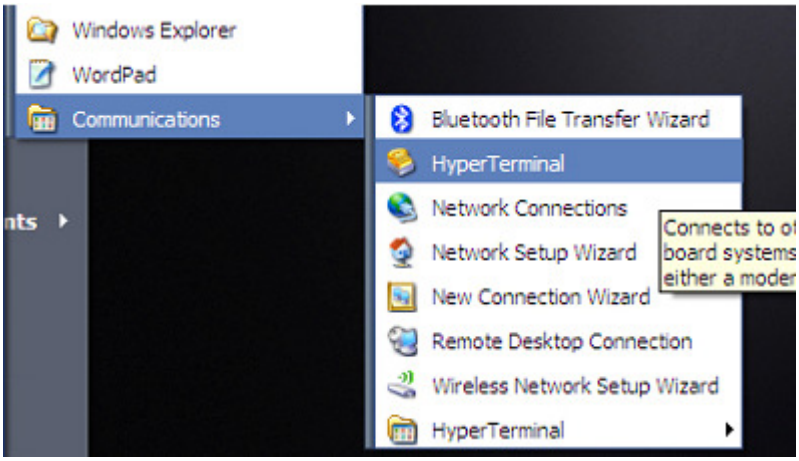
4. Using UART

a) Check push button (must be "low" for UART)

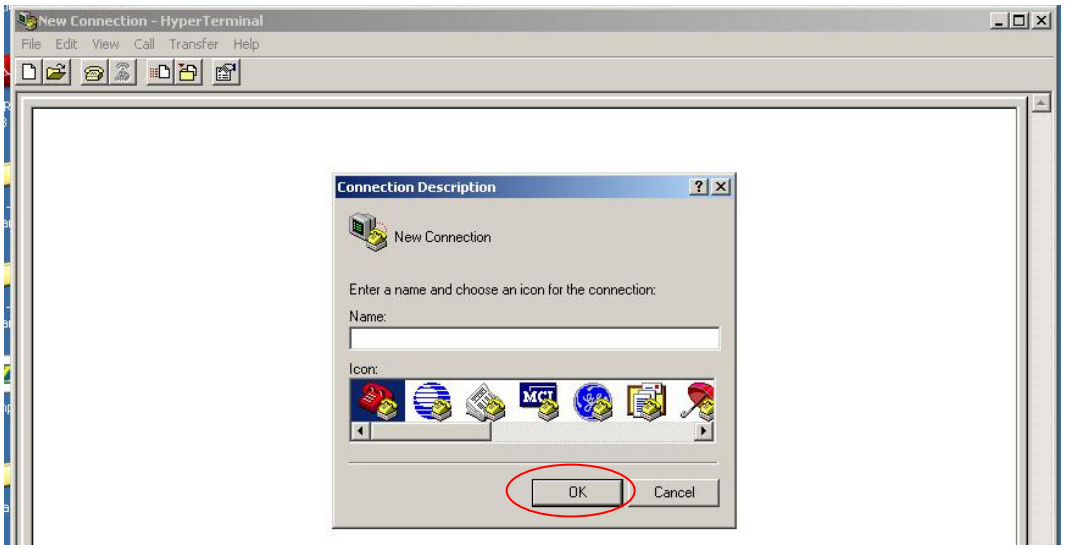
● *Look for the toggle button near the serial port on the SlimBoard. This is the selector for the DEBUG/UART function of the serial port.*



b) Open **HyperTerminal**



c) Enter name of new connection and click **OK**.



d) Select correct Com Port on the **Connect Using** field.

e) Click **OK**

f) Match this data:

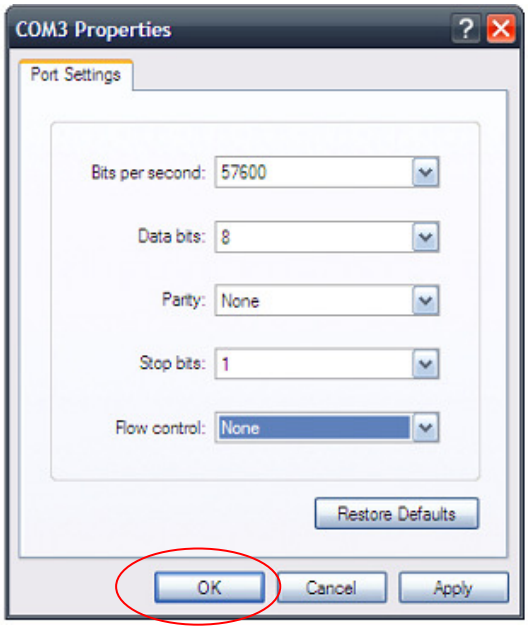
i) **Bits per second: 57600**

ii) **Databits: 8**

iii) **Parity: none**

iv) **Stop bits: 1**

v) **Flow control: none**

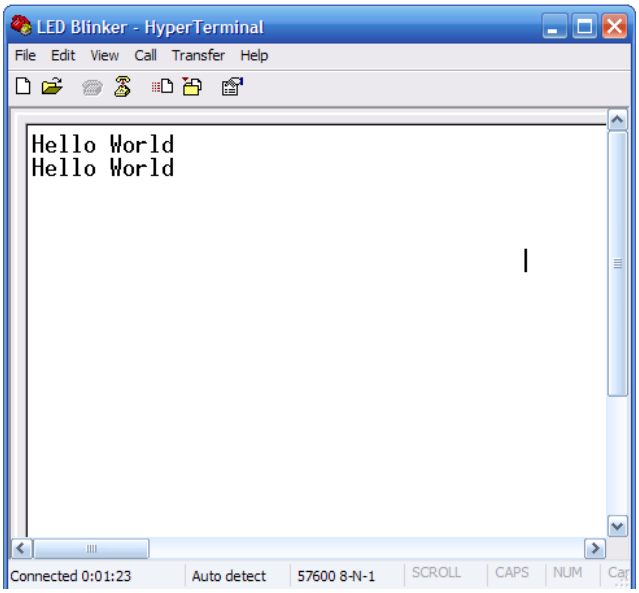


g) Click **OK**

● *After clicking ok, the hyperterminal will automatically place a call. But if it doesn't, click the Call button on the toolbar to place a connection.*

h) **Reset** SlimBoard

i) Check the **Hyperterminal** console, “Hello World” should appear once:



- *When you are done testing, you can end the call by clicking the **Disconnect** button on the toolbar.*

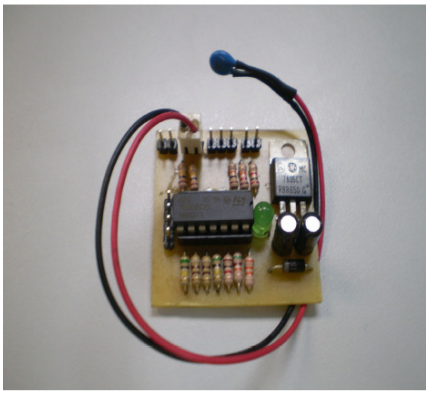
B. Sample LED Blinker Code

```
/* This is includes the three header files namely: ez8.h,stdio.h,sio.h. The
header files contain the necessary functions that is used in microcontrollers.
*/
#include <ez8.h>
#include <stdio.h>
#include <sio.h>

/*Defines that all instances of GPIO_PORT be replaced by 0x03 */
#define GPIO_PORT 0x03

/*This is called a function. This creates a function named initGPIO */
void initGPIO(void) {

	PDDD &= ~GPIO_PORT;		// data direction: output
	PDAF &= ~GPIO_PORT;		// alternate function: off
//	PDOC |= GPIO_PORT;		// output control: open-drain (optional)
//	PDHDE |= GPIO_PORT;		// high drive enable: high drive
					//(optional)
}

/* This creates a function named delay. Essentialy, this function counts from 0
to FF many times. This causes some processing time delay. */
void delay(void)
{
	unsigned char i,j;

	for(i = 0; i < 0xFF; i++)
		for(j = 0; j < 0xFF; j++);
}

`	/* This is the main function. This is the function that is executed first. */
int main(void)
{
	init_uart(_UART1, _DEFFREQ, _DEFBAUD); // Initializes UART
	initGPIO();  // Calls the function that we created earlier

	printf("Hello World\n");  /*printf is a function that displays what is
				  inside its parameters. You could try to replace
				  “Hello World” with something else */

	while(1)	// essentially a never-ending loop
	{
		PDOUT ^= GPIO_PORT; /* Compares the current value of PDOUT with the
				       value of GPIO_PORT using XOR. */
		delay();	/* Calls the function delay. This causes the LED to
				   blink */
	}
}
```

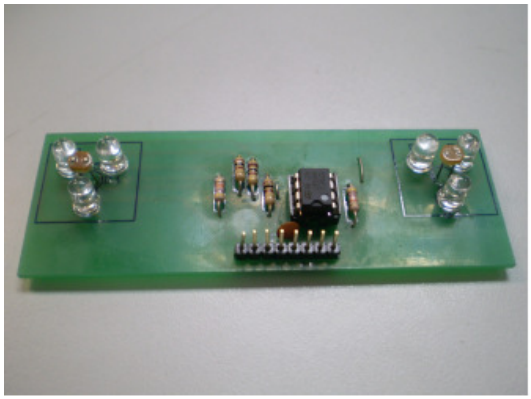
ADDITIONAL INFORMATION

For more advanced projects, you could buy the IRC Microcontroller Training kit. This package contains the following components:

a. UP IRC SlimBoard with Z8F6421 chip
b. DC Adapter
c. Installation and Documentation CD
d. Serial Cable
e. SlimBoard Workbook
f. Protoboard/Breadboard
g. Pre-stripped wires and Alligator Clips
h. 9-V battery
i. Colorimeter Module
j. Temperature Signal Conditioning Module
k. Servomotor
l. LED Set (R, G, B or Y)
m. Piezo-electric Buzzer
n. Resistors, LDR, Potentiometer



Temperature Signal Conditioning Module



Colorimeter Module

UP IRC Microcontroller Training Kit

For inquiries, you could call the **Instrumentations, Robotics, and Control Laboratory** at:
Tel. no: (+632)981-8500 loc. 3360
Fax no: (+632)925-2957

INSTRUMENTATION, ROBOTICS, AND CONTROL LABORATORY

Room 311, Department of Electrical and Electronics Engineering

DEEE Building, Velasquez Street, U.P. Campus, Diliman, Quezon City